THESIS FOR THE DEGREE OF LICENTIATE OF SCIENCE

Efficient Approximate Big Data Clustering: Distributed and Parallel Algorithms in the Spectrum of IoT Architectures

AMIR KERAMATIAN

Division of Networks and Systems
Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Efficient Approximate Big Data Clustering: Distributed and Parallel Algorithms in the Spectrum of IoT Architectures
Amir Keramatian

Copyright © Amir Keramatian, 2019.

Technical report 207L
ISSN 1652-876X
Department of Computer Science and Engineering
Distributed Computing and Systems Group

Division of Networks and Systems
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31-772 10 00

Author e-mail: amirke@chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Efficient Approximate Big Data Clustering: Distributed and Parallel Algorithms in the Spectrum of IoT Architectures

Amir Keramatian

Division of Networks and Systems, Chalmers University of Technology

ABSTRACT

Clustering, the task of grouping together similar items, is a frequently used method for processing data, with numerous applications. Clustering the data generated by sensors in the Internet of Things, for instance, can be useful for monitoring and making control decisions. For example, a cyber physical environment can be monitored by one or more 3D laser-based sensors to detect the objects in that environment and avoid critical situations, e.g. collisions.

With the advancements in IoT-based systems, the volume of data produced by, typically high-rate, sensors has become immense. For example, a 3D laser-based sensor with a spinning head can produce hundreds of thousands of points in each second. Clustering such a large volume of data using conventional clustering methods takes too long time, violating the time-sensitivity requirements of applications leveraging the outcome of the clustering. For example, collisions in a cyber physical environment must be prevented as fast as possible.

The thesis contributes to efficient clustering methods for distributed and parallel computing architectures, representative of the processing environments in IoT-based systems. To that end, the thesis proposes MAD-C (abbreviating Multi-stage Approximate Distributed Cluster-Combining) and PARMA-CC (abbreviating Parallel Multiphase Approximate Cluster Combining). MAD-C is a method for distributed approximate data clustering. MAD-C employs an approximation-based data synopsis that drastically lowers the required communication bandwidth among the distributed nodes and achieves multiplicative savings in computation time, compared to a baseline that centrally gathers and clusters the data. PARMA-CC is a method for parallel approximate data clustering on multi-cores. Employing approximation-based data synopsis, PARMA-CC achieves scalability on multi-cores by increasing the synergy between the work-sharing procedure and data structures to facilitate highly paral-

lel execution of threads. The thesis provides analytical and empirical evaluation for MAD-C and PARMA-CC.

Keywords: Approximation-based synopsis, Clustering, Distributed and Parallel Processing

Preface

Parts of the contributions presented in this thesis have previously appeared in the following manuscript.

- **Amir Keramatian**, Vincenzo Gulisano, Marina Papatriantafilou, Philippas Tsigas, and Yiannis Nikolakopoulos
  "MAD-C: Multi-stage Approximate Distributed Cluster-combining for Obstacle Detection and Localization"
  Appeared in the *F2C-DP workshop in the 24th European Conference on Parallel Processing, 2018*. In addition, an extended and revised version is submitted for journal publication.

- **Amir Keramatian**, Vincenzo Gulisano, Marina Papatriantafilou, and Philippas Tsigas
  "PARMA-CC: Parallel Multiphase Approximate Cluster Combining"
  To appear in the *21st International Conference on Distributed Computing and Networking, 2020*.

Acknowledgments

I am grateful for the invaluable guidance and support I receive from my supervisors Marina Papatriantafilou, Vincenzo Gulisano, and Philippas Tsigas. I thank Yiannis Nikolakopoulos for having done collaboration with me. Furthermore, I would like to thank Agneta Nilsson and Tomas Olovsson for the time and effort they have put for helping me.

Colleagues of mine have made the experience of working at Chalmers even more pleasurable. Many thanks to Adones, Ali, Aljoscha, Aras, Bastian, Bei, Beshr, Boel, Carlo, Charalampos, Christos, Dimitris, Elad, Elena, Fazeleh, Francisco, Georgia, Hannah, Iosif, Ismail, Ivan, Joris, Karl, Katerina, Nasser, Oliver, Romaric, Thomas, Vladimir and Wissam.

I would like to acknowledge the financial support by the Swedish Foundation for Strategic Research, project Factories in the Cloud (FiC), with grant number GMT14-0032.

Here is a good place as any to express my gratitude towards my parents. My father inspired me with innovative thinking, and my mother taught me to aim high. Besides family, my friends have generously provided support and care for me. By this means I would like to thank Mozhgan, Elin, Sharon, and Hamid.

Amir Keramatian
Göteborg, December 2019

Contents

List of Figures

Part I

OVERVIEW

1 Overview

1.1 Introduction

Devices in *IoT*, Internet of Things, are changing our everyday lives by making everything connected and automated to a certain degree. Ericsson forecasts the number of IoT devices to hit around 20 billion by 2022. The *thing* in IoT is an entity with an embedded system that has the ability to transfer data over a network. For example, it can be a heart monitor and the associated communicating devices inside a person's body, or it can be a vehicle with tens of processing units and sensors on board.

An IoT-based system typically consists of a network of cyber-physical systems (e.g. a variety of sensors, smart meters, actuators, pumps, and engines), resource-constrained computational devices (e.g. handheld mobile computers), high-end servers, communication medium and other associated devices.

IoT has been effectively incorporated in industrial use-cases [1]. For example, in the fourth industrial revolution [2], the production facilities in a factory, such as Automated Guided Vehicles (AGV) and robot arms, are managed, controlled, and optimized using IoT-based systems.

The following discusses a three-level computational architecture that facilitates connection and computing in IoT-based systems.

A three-level computational architecture: cloud, fog, and edge

It is a common practice to transfer data into the *cloud*, where storage, computation, analysis, and decision making take place [3]. As the typical infrastructure in cloud provides abundant resources (e.g. powerfull high-end servers), *cloud computing* can be used to solve problems in IoT-based systems. Cloud computing has several advantages. Firstly, computational scalability can be achieved by properly leveraging the resources of high-end servers. Secondly, the cloud services typically enable the users to pay for the services in a pay-as-you-go manner, making it more economical than having on-premise equipment. Thirdly, the cloud is maintained by the cloud service providers, sparing the end users from technical managements. These advantages show the importance of cloud computing for many problems in IoT-based systems.

On the other hand, cloud computing has some disadvantages as well. Considering the distance between the devices and cloud service providers, the services might suffer from high latency. Moreover, transmitting the data from all the devices in the network requires high bandwidth. Furthermore, security measures are required to transmit all the raw data to a cloud service provider.

Fog and *edge* computing have become popular in deployment to address the concerns regarding cloud computing. Edge computing refers to the computation, analysis, storage, and decision making that takes place at *the edge of the network*, i.e. the devices (like mobile phones) directly connected to the sensors or an IoT gateway closely located to the sensors. *Fog computing* refers to leveraging the computational and storage capacities within a local network of systems (e.g. a LAN) to carry out the computation that would, otherwise, require the resources of a cloud infrastructure. Compared to the cloud computing, fog and edge computing facilitate processing data

closer to where it is produced [3]. As a result, adapting IoT-based systems to fog/edge computing can provide solutions with lower latency and higher security, as the data remains within the local network. Nevertheless, fog/edge computing can be challenging as fog/edge devices are typically resource-constrained in terms of computational power. Furthermore, such devices can be connected via limited bandwidth media, e.g. wireless networks, which makes transmitting large volumes of data expensive.

1.2 The scope of this thesis

This thesis revolves around methods for efficient data processing using powerful high-end servers (typically deployed in the cloud computing), and resource-constrained devices (typically deployed in fog/edge computing). The following subsections introduce the different aspects of the thesis.

1.2.1 Processing sensor data

Sensors installed in IoT devices continuously gather data as they take measurements in their environment. Based on the latent information in the sensors' data, new control decisions might be taken. For example, when the data from smoke and temperature sensors indicate a possibility of fire, an alarm should go off.

With the advancements in IoT-based systems, modern sensors produce *large volume* of data in a *high velocity*. Furthermore, the data might be generated from a *variety of sources*. These aspects are reminiscent of the *3V* model [4] for big data.

Processing big data with conventional methods is challenging. Basically, processing a too large volume of data with a conventional data processing method requires excessive amounts of time which can be detrimental to time-sensitive problems, e.g. navigating a self driving vehicle. Moreover, processing data that is produced at high rates requires methods that can perform the processing in a single pass and on the fly. Lastly, processing the *fused data*, aggregated data that is generated from multiple sources, requires methods capable of processing heterogeneous data. For example, an AGV, for the purpose of navigation and avoiding obstacles, might be equipped with high-rate laser-based sensors, generating a high volume of data every second,

digital cameras, and various other sensors. The following outlines the methods that this thesis employs to address the challenges in processing big data.

1.2.2 Methods for dealing with big data produced by sensors

To address the challenges of processing big data, the methods proposed in this thesis use the insights from [5]: (i) *scaling down* the amount of data to be processed, (ii) utilizing parallel processing and efficient use of shared memory to *scale up* the computing on a node, and (iii) utilizing distributed nodes (e.g. in fog/edge) to *scale out* the computing to different nodes.

Scaling down the data

The first line of defence against big data is scaling down the large volume of data. Scaling down can be applied in a variety of forms [6], for instance, sampling [7, 8], dimensionality reduction [9], and compression [10].

Most applications do not require exact solutions; however, they need to make decisions as fast as possible. For instance, regarding the problem of managing AGVs, a real-time solution that approximately detects the objects in the environment and avoids hitting them is more helpful than an exact solution that detects the objects with all the details but fails to to finish within reasonable amount of time, leading to catastrophic outcomes. Therefore, approximation is an important tool for scaling down the volume of data in order to achieve timely solutions.

Approximation techniques can be employed to derive a data synopsis orders of magnitude smaller than the original data, leading to orders of magnitude speed-up in processing time. Often, it is important that the data synopsis be constructed in only one pass over the original data. Furthermore, as the data might not be present all at once and become available gradually (for instance the manner in which most sensors provide readings), the data synopsis can be constructed incrementally, with constant overhead per new data entry. *Sketches* [11] are important approximation-based synopsis for big data.

Scaling up the computing in one node

Modern computing platforms support concurrent execution, synchronization, and communication of many workers (e.g. threads, CPUs, etc), sharing the available resources of the system, for instance memory. Leveraging parallel processing and efficient use of the shared memory can lead to scaling up the computing on a single node. Maximizing such scalability is particularly important on powerful high-end servers (e.g. those typically employed in the cloud computing) as they provide abundant resources in terms of computational power, memory, and storage. For example, a high-end server can consist of several sockets, each socket accommodating several cores, and each core can possibly support hyper-threading [12].

However, regarding many problems in IoT-based systems, achieving and maintaining high scalability with increasing number of workers is challenging. In fact, increasing the number of workers might even reduce the scalability of a system. The following presents a discussion on the affiliated challenges with scalability of a system considering varying number of workers.

The Universal Scalability Law. USL, or the universal scalability law [13], proposed by Neil Gunther, is a model to quantify the scalability of a system. According to USL, there are two main obstacles on the way of achieving linear scalability with the number of workers. The first one is *contention*, and the second one is *crosstalk* [14]. Contention happens when the workers of a task require a shared resource but can not have it at the same time. Therefore, they have to queue up, which negatively affects the parallelization of the task. Crosstalk happens when the pairs of workers need to communicate in order to synchronize and share their states, which negatively affects the scalability. Based on contention and crosstalk, USL models the scalability of a system as the following:

$$S_K = \frac{K}{1 + \delta(K-1) + \kappa K(K-1)},$$

where S_K is the scalability with K workers, and δ is the contention degree coefficient, and κ is the crosstalk penalty coefficient. Note δ gets multiplied by $(K-1)$ which shows contention grows linearly in the number of workers (i.e. as they queue up for

a shared resource). On the other hand, κ gets multiplied by $K(K-1)$ reflecting the crosstalk between all pairs of workers in the system. Note that, in an ideal system, coefficients δ and κ are zero; therefore, the scalability of the system increases linearly with the number of workers. Nevertheless, with κ and δ being greater than zero, the growth of denominator gets higher than the numerator, and S_K will start to decrease after a large enough value of K.

Therefore, to achieve high scalability, the challenge is to design data structures, algorithms, and work sharing schemes that minimize the contention for the shared resources and the crosstalk among the pairs of workers.

Scaling out the computing to distributed nodes in fog/edge

Increasing the number of computing nodes is another approach to deal with big data [5, 15, 16]. To that end, an interconnected network of fog/edge devices (with attached sensors) can be leveraged to process the gathered sensor data. As the communication takes place in the local network of the devices, the communication latency is lower than the cloud computing setup, and the privacy of data is preserved within the boundaries of the local network.

Nevertheless, there are two challenges that need to be considered in designing an IoT-based system that scales out the computing. The first one concerns the fact that the fog/edge devices are weaker than the high-end servers, in the sense that they have limited computational and memory capacities. The second challenge arises from the limited-bandwidth shared communication medium that interconnects the fog/edge devices.

1.2.3 Algorithmic engineering concerns

To fully exploit the potential computational capacity of a hardware platform, this thesis considers (algorithmic) implementation details to maximize the synergy between software and hardware [17]. In other words, this thesis takes into account concerns regarding the design and algorithmic implementation of the solutions based on the behaviour of the target platform. For example, regarding the high-end servers supporting several workers, solutions must result in minimum contention and crosstalk

among the workers, as suggested by USL. Similarly, regarding the resource-constrained devices in fog/edge computing, solutions must be designed in order to minimize the communication volume among the fog/edge devices, and perform the computational tasks as close as possible to the generating sources of data.

1.3 Preliminaries

This section introduces the required background for reading the following sections.

1.3.1 LIDAR and point clouds

LIDAR (LIght Detection And Ranging) is a scanning method to generate a 3-D representation of a target by illuminating the target with pulsed light waves. The 3-D model is generated based on the time that light waves take to return.

A LIDAR scanner typically leverages several laser beams (e.g. 8 to 128). Located on the spinning head of the sensor, each laser beam emits photons and reads back their reflection several times in a second. As the spinning head of the LIDAR scanner makes a full rotation, a 360 degree high resolution 3-D representation of the environment is attained [18].

The readings from a LIDAR sensor are named *point clouds* [19]. Point clouds generated by certain types of LIDAR sensors can contain hundreds of thousands of 3-D points, or even more. The point cloud from a full rotation of a LIDAR sensor's spinning head contains points corresponding to *scene objects* (e.g. pedestrians, vehicles, etc) in the environment in which the LIDAR sensor is installed.

1.3.2 Cluster Analysis

Cluster analysis, or clustering, is the task of partitioning a given set of items into clusters, where within a cluster items are more *similar* to each other than to those in other clusters [20, Ch. 11-16]. More specifically, the task is to assign the same clustering *label* to the items in the same cluster, but different from those in other clusters.

Cluster analysis is an important data analysis tool. Certain types of clustering algorithms can be used to segment/partition the points in a point cloud based on the scene object to which they belong. In other words, the clustering labels can differentiate the points based on the scene object that each point is a member of [21, 22]. Cluster analysis can also be applied on GPS data to understand mobility and route analysis [23, 24]. The two types of clustering algorithms that this thesis concerns are introduced in the following paragraphs.

Distance-based clustering. The items grouped together in a distance-based cluster satisfy some distance-related criteria. For example, the Euclidean clustering algorithm [21, 22] partitions a given point cloud into an a priori unknown number of clusters. It works with two adjustable parameters: `minPts` and ϵ. The algorithm produces clusters each containing at least `minPts` number of points, and within each cluster, each point lies in the ϵ-radius neighbourhood of at least another point in the same cluster. Points not belonging to any cluster are characterized as *noise*. Lisco [25] is a single-pass continuous version of the Euclidean clustering algorithm designed for sorted data; e.g. a point cloud gathered by a LIDAR sensor is *angularly* sorted because the point cloud is collected by the spinning head of the sensor.

Density-based clustering. The density-based clustering algorithms partition a given point-cloud into high density regions (clusters), separated by contiguous regions of low density regions. For example, Density-Based Spatial Clustering of Applications with Noise, or DBSCAN [26], is a *density-based* clustering algorithm that partitions a given point-cloud into an a priori unknown number of clusters. Similar to the Euclidean clustering algorithm, DBSCAN works with two adjustable parameters: `minPts` and ϵ. A Cluster found by DBSCAN consists of at least one *core point* and all the points that are *density-reachable* from it. Point p is a core point if it has at least `minPts` points in its ϵ-radius neighbourhood. Point q is *directly reachable* from p if q lies in the ϵ-radius neighbourhood of p. Point q is density-reachable from p, if q is directly reachable either from p or another core point that is density-reachable from p. Non-core points that are not density-reachable from any core-points are outliers [27, 28]. DENCLUE [29], STING [30], and OPTICS [31] are some other well-known

density-based clustering algorithms.

1.4 Research Challenges

This thesis studies the problem of efficient sensor data clustering (e.g. point clouds or GPS readings) from one or more sources as a representative problem in IoT-based systems.

The aforementioned clustering task on point clouds can be used for extracting valuable information regarding the objects and their dispositions in the environment scanned by one or more LIDAR sensors. Considering the problem of managing AGVs equipped with LIDAR sensors, clustering point clouds can be used for detection and localization of scene-objects. Furthermore, it can be used for online monitoring of the environment, guiding the AGV through the environment, and making sure that it does not collide with other objects. It can also be employed to make sure that the AGV does not enter a restricted area in the environment, framed by a *geofences*.

To counter-balance the overwhelming effects of the big data produced by sensors (e.g. LIDAR sensors), the methods in this thesis leverage scaling down, scaling up, and scaling out (discussed in § 1.2) techniques. § 1.4.1 characterizes the distributed version of the problem, which employs the scaling down and scaling out techniques. § 1.4.2 studies the problem using parallel computing, which the employs scaling down and scaling up techniques. § 1.6 summarizes the contributions of the thesis regarding each variant of the problem.

1.4.1 Clustering Sensor Data Distributedly Gathered by Multiple Fog/Edge Devices

Combining readings from multiple sensors is commonly recognized as *sensor fusion*. In certain scenarios it is useful to have *multiple* sensors scan the environment simultaneously. For example, a LIDAR sensor installed in a particular place in the environment might have an *occluded* view. In order to address the occlusion [32] and increase safety and robustness, several LIDAR sensors can be installed in different

locations to scan the environment from different perspectives. To that end, the first direction of research that this thesis follows regarding the problem of efficient sensor data clustering is leveraging a network of fog/edge devices with attached sensors for distributed clustering of the sensor data, i.e. to scale out as noted in § 1.2.2.

Challenges. Centrally gathering all the local point clouds into a single source and performing the clustering algorithm on their union, known as the *merged point cloud*, is cumbersome. Firstly, transmitting several point clouds (each containing hundreds of thousands of points) imposes high latency and takes long time, as long as a few tens of seconds. The latter becomes even more challenging when the devices use the wireless medium for communication, which has a low bandwidth and has a high risk of transmission failures. Secondly, computationally speaking, performing the clustering algorithm on the merged point cloud takes excessive amount of time on the fog/edge devices as there can be many millions of points in the merged point cloud.

Research Questions. The aforementioned challenges arise from centrally gathering all the point clouds. On the other hand, considering the distributed nature of the problem, can processing the data closer to the generating sources increase the computational efficiency and lower the communication overhead (Q_1)? Moreover, as mentioned in § 1.2, one primary technique to overcome the huge volume of data is to first scale it down. How can approximation-based synopsis help to reduce the required communication and computational resources (Q_2)? This thesis introduces a solution that jointly answers Q_1 and Q_2.

1.4.2 Highly Parallel Clustering on Shared Memory Multi-Core Systems

The second direction of research that this thesis follows regarding the problem of efficient sensor data clustering is leveraging the computational resources of a high-end server for parallel clustering of sensor data, i.e. to scale up [5] on a single node, as noted in § 1.2.2.

Challenges. A parallel clustering algorithm employing several workers imposes the following challenges: (i) the distribution of the workload among the workers, (ii) how each worker processes its assigned workload, and (iii) synchronization and communication among the workers. In order to increase the scalability of the algorithm, the aforementioned challenges must be addressed in a way to decrease the contention for the shared resources and the crosstalk among each pair of workers, as suggested by USL, see § 1.2.2.

Regarding point clouds, the majority of commonly used data structures (e.g., KD-trees, Octrees [33], and voxel grids [21]), grow in size with respect to the size of the point cloud. Furthermore, the cost of common queries and operations can become expensive (e.g. super linear in the number of data points). Moreover, such data structures show their worst-case performance when the distribution of data is skewed [34]. Therefore, processing point clouds and GPS readings is challenging with such data structures.

Research Questions. The work-sharing design of a parallel clustering algorithm distributes the work-load among the workers and regulates the processing of each worker, as well as the communication and synchronization among the workers. Considering USL, can the synergy from the joint design of the work-sharing mechanism and the data structures increase the scalability by lowering crosstalk among the pairs of workers and the contention for the shared data (Q_3)? Moreover, can the total amount of work-load be reduced by incorporating approximation-based synopsis into the data structures, thus increasing scalability (Q_4)? Furthermore, can leveraging the approximation-based synopsis achieve *super-linear* scalability in the number of workers (Q_5)? This thesis studies how questions Q_3, Q_4, and Q_5 can be jointly answered.

1.5 Contributions

This section outlines the contributions of the thesis regarding the research problems in § 1.4.

1.5.1 Multi-stage Approximate Distributed Cluster-combining for Obstacle Detection and Localization (Paper I)

Regarding the problem of distributed sensor data clustering on fog and edge devices, this thesis, in Chapter 2, proposes MAD-C, abbreviating Multi-stage Approximate Distributed Cluster-combining for Obstacle Detection and Localization.

In MAD-C, the devices, distributedly and in parallel, perform the substantial portion of the required processing locally, addressing research question Q_1 in § 1.4.1. Furthermore, employing an approximation-based synopsis, each device locally approximates the results of its local processing. Afterwards, the fog/edge devices communicate and combine the approximation-based synopses in order to make a global synopsis corresponding to the clustering of the merged point cloud. Each fog/edge device can generate its local approximate-based synopsis incrementally with only one pass over the data. As the local synopses are much smaller in size (compared to the original local point clouds) and can be combined efficiently, MAD-C achieves significant communication and computational savings, addressing research question Q_2 in § 1.4.1. In addition to analytical results, empirical evaluation of MAD-C, both in simulation and a setup of fog/edge devices), shows advantages of MAD-C regarding Q_1 and Q_2.

Advances relative to the state of the art. Regarding the problems that concern multiple sources of point clouds, the common practice is to perform processing on the merged point clouds, for example [35]. Nevertheless, as pointed out in § 1.4.1, when the data is gathered distributedly, centrally gathering and clustering the point clouds is inefficient. In such cases, distributed clustering algorithms can be leveraged. For example, DBDC [36], density-based distributed clustering, is a client-server method that approximates the clustering outcome of DBSCAN. Client nodes in DBDC locally perform operations on their data and transmit some representatives, on which the server performs some extra processing and forwards the results back to the clients. Unlike MAD-C's constant-size approximation-based synopsis, there are no guarantees on the number of representatives in DBDC. Furthermore, MAD-C nodes can operate with an arbitrary connection topology. Authors in [37] propose

distributed versions of some center-based clustering algorithms (e.g. K-Means, K-Harmonic-Means, and Expectation-Maximization). In an iterative approach, each local node computes sufficient statistics for its local data and then receives and aggregates sufficient statistics from other nodes to attain a global sufficient statistics. Nevertheless, MAD-C is more efficient at communication as the transmission of data between MAD-C nodes takes place only once.

1.5.2 Parallel Multistage Approximate Cluster Combining (Paper II)

Regarding the problem of parallel sensor data clustering on high-end servers, this thesis, in Chapter 3, proposes PARMA-CC, Parallel Multistage Approximate Cluster Combining. PARMA-CC facilitates leveraging an arbitrary number of workers to achieve a high degree of scalability. It employs the same approximation-based synopsis developed for MAD-C to scale down the large volume of data.

In PARMA-CC, the synergy between the data structures and the work sharing schemes reduce the contention for the shared resources and the crosstalk between pairs of workers, as suggested by USL. More specifically, PARMA-CC leverages data-parallelism and a specially designed shared data structure that supports in-place operations (eliminating the need for data copying or migration). Moreover, the work-sharing mechanism regulates the workers to behave in a divide-and-conquer manner, reducing the communication and synchronization among the workers. It also controls how the workers access the shared data structure in order to avoid contention. Exploiting the aforementioned contributions, PARMA-CC addresses the research question Q_3 in § 1.4.2.

Furthermore, the empirical results of PARMA-CC show that the total amount of work-load can be reduced by incorporating the approximation-based synopsis, addressing the research question Q_4 in § 1.4.2. In addition, when the input data has a skewed distribution, PARMA-CC can achieve super-linear scalability in the number of workers, addressing the research question Q_5 in § 1.4.2. The thesis provides analytical and empirical results to show the latter.

Advances relative to the state of the art. There are three relevant categories of related work to PARMA-CC. The first category consists of distance-based and density-based clustering methods, introduced in § 1.3. PARMA-CC can approximate the clustering outcome of many distance-based and density-based methods through its approximation-based synopsis. The second category contains methods that boost the performance of conventional density-based and distance-based clustering algorithms via parallelization. For example, Highly Parallel DBSCAN [38], HPDBSCAN, is an OpenMP/MPI hybrid algorithm. HPDBSCAN offers good scalability; however, when the data is skewed, its performance degrades severely. On the other hand, PARMA-CC better tolerates skewed data. Moreover, the way that PARMA-CC is designed to utilize the shared memory via in-place operations is more efficient than OpenMP's relaxed consistency memory model in which multiple copies of the same data might exist. The last category consists of methods that, through approximation, sacrifice accuracy to gain performance. For example, ρ-approximate DBSCAN [28], and STING [30] approximate the clustering outcome of DBSCAN. Other approximation approaches employ sampling techniques, e.g. [39]. These approaches can as well be utilized in PARMA-CC's approximation-based synopsis.

1.6 Conclusions and Future Work

This thesis studies the problem of efficient sensor data clustering from one or more sources employing powerful high-end servers or resource-constrained fog/edge devices. Considering the large volume, velocity, (and possibly variety) of the data to be clustered, this thesis proposes an approximation-based data synopsis to scale down the data. Furthermore, it proposes methods tailored for each type of environment (i.e. high-end servers or fog/edge devices) to efficiently address the problem using the approximation-based data synopsis.

The methods introduced in this thesis can form a basis for a more general data processing framework. A future line of research can fuse data from other types of sources (e.g. positioning sensors) with LIDAR data, in order to increase accuracy and safety. Another future line of research is to extend the methods to predict the *type* (e.g. pedestrian, cyclist, etc) of the scene-objects. Moreover, the methods can be

extended to be employed in a continuous fashion. As the readings from consecutive rotations of a LIDAR sensor are similar, it is expected that an approximation-based synopsis corresponding to one rotation of a LIDAR sensor can be employed to efficiently generate synopsis corresponding to the next rotation of the LIDAR sensor.

Bibliography

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov 2014.

[2] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, Aug 2014.

[3] Mung Chiang and Tao Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[4] Todor Ivanov, Nikolaos Korfiatis, and Roberto V. Zicari, "On the inequality of the 3v's of big data architectural paradigms: A case for heterogeneity," *CoRR*, vol. abs/1311.0805, 2013.

[5] P. B. Gibbons, "Big data: Scale down, scale up, scale out," in *2015 IEEE International Parallel and Distributed Processing Symposium*, May 2015, pp. 3–3.

[6] Muhammad Habib ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman, Teh Ying Wah, and Samee U. Khan, "Big data reduction methods: A survey," *Data Science and Engineering*, vol. 1, no. 4, pp. 265–284, Dec 2016.

[7] Mohamed Medhat Gaber, Arkady B. Zaslavsky, and Shonali Krishnaswamy, "Mining data streams: a review," *SIGMOD Record*, vol. 34, no. 2, pp. 18–26, 2005.

[8] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman, *Mining of Massive Datasets, 2nd Ed*, Cambridge University Press, 2014.

[9] Carlos Oscar Sánchez Sorzano, Javier Vargas, and Alberto Domingo Pascual-Montano, "A survey of dimensionality reduction techniques," *CoRR*, vol. abs/1403.2877, 2014.

[10] B. Havers, R. Duvignau, H. Najdataei, V. Gulisano, A. C. Koppisetty, and M. Papatriantafilou, "Driven: a framework for efficient data retrieval and clustering in vehicular networks," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, April 2019, pp. 1850–1861.

[11] Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends in Databases*, vol. 4, no. 1-3, pp. 1–294, 2012.

[12] Avinash Sodani, Roger Gramunt, Jesús Corbal, Ho-Seop Kim, Krishna Vinod, Sundaram Chinthamani, Steven Hutsell, Rajat Agarwal, and Yen-Chen Liu, "Knights landing: Second-generation intel xeon phi product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016.

[13] Neil J. Gunther, "A general theory of computational scalability based on rational functions," 2008.

[14] B Schwarz, "Practical scalability analysis with the universal scalability law," 2015.

[15] Nawsher Khan, Arshi Naim, Mohammad Rashid Hussain, Noorulhasan Naveed Quadri, Naim Ahmad, and Shamimul Qamar, "The 51 v's of big data: Survey, technologies, characteristics, opportunities, issues and challenges," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems, COINS 2019, Crete, Greece, May 5-7, 2019*, Farshad Firouzi, Krishnendu Chakrabarty, Bahar Farahani, Fangming Ye, and Vasilis F. Pavlidis, Eds. 2019, pp. 19–24, ACM.

[16] Raja Appuswamy, Christos Gkantsidis, Dushyanth Narayanan, Orion Hodson, and Antony I. T. Rowstron, "Scale-up vs scale-out for hadoop: time to rethink?," in *ACM Symposium on Cloud Computing, SOCC '13, Santa Clara, CA, USA, October 1-3, 2013*, Guy M. Lohman, Ed. 2013, pp. 20:1–20:13, ACM.

[17] G. De Michell and R. K. Gupta, "Hardware/software co-design," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 349–365, March 1997.

[18] Gerardo Atanacio-Jimenez, Jose-Joel Gonzalez-Barbosa, Juan B. Hurtado-Ramos, Francisco J. Ornelas-Rodriguez, Hugo Jimenez-Hernandez, Teresa Garcia-Ramirez, and Ricardo Gonzalez-Barbosa, "Lidar velodyne hdl-64e calibration using pattern planes," *International Journal of Advanced Robotic Systems*, vol. 8, no. 5, pp. 59, 2011.

[19] Brent Schwarz, "Lidar: Mapping the world in 3d," *Nature Photonics*, vol. 4, no. 7, pp. 429, 2010.

[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Elsevier Science, 2008.

[21] Radu Bogdan Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," *KI - Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, Nov 2010.

[22] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[23] Radu Mariescu-Istodor and Pasi Fränti, "Grid-based method for gps route analysis for retrieval," *ACM Transactions on Spatial Algorithms and Systems*, vol. 3, no. 3, pp. 8, 2017.

[24] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, New York, NY, USA, 2008, UbiComp '08, pp. 312–321, ACM.

[25] H. Najdataei, Y. Nikolakopoulos, V. Gulisano, and M. Papatriantafilou, "Continuous and parallel lidar point-cloud clustering," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, July 2018, pp. 671–684.

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996, KDD'96, pp. 226–231, AAAI Press.

[27] Junhao Gan and Yufei Tao, "Dbscan revisited: Mis-claim, un-fixability, and approximation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2015, SIGMOD '15, pp. 519–530, ACM.

[28] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 19:1–19:21, July 2017.

[29] Alexander Hinneburg, Daniel A Keim, et al., "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, 1998, vol. 98, pp. 58–65.

[30] Wei Wang, Jiong Yang, and Richard R. Muntz, "Sting: A statistical information grid approach to spatial data mining," in *Proceedings of the 23rd International Conference on Very Large Data Bases*. 1997, VLDB '97, pp. 186–195, Morgan Kaufmann Publishers Inc.

[31] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander, "Optics: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 1999, SIGMOD '99, pp. 49–60, ACM.

[32] Niklas Elmqvist and Philippas Tsigas, "A taxonomy of 3d occlusion management for visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1095–1109, Sept. 2008.

[33] Jan Elseberg, Dorit Borrmann, and Andreas Nuchter, "One billion points in the cloud - an octree for efficient processing of 3d laser scans," *International Journal of Photogrammetry and Remote Sensing*, vol. 76, pp. 76–88, 02 2013.

[34] Wei Wang, Jiong Yang, and Richard Muntz, *PK-Tree: A Spatial Index Structure for High Dimensional Point Data*, pp. 281–293, Springer US, Boston, MA, 2000.

[35] Muhammad Sualeh and Gon-Woo Kim, "Dynamic multi-lidar based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, pp. 1474, 2019.

[36] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle, "Towards effective and efficient distributed clustering," in *In Workshop on Clustering Large Data Sets (ICDM)*, 2003, pp. 49–58.

[37] George Forman and Bin Zhang, "Distributed data clustering can be efficient and exact," *SIGKDD Explorations*, vol. 2, no. 2, pp. 34–38, 2000.

[38] Markus Götz, Christian Bodenstein, and Morris Riedel, "Hpdbscan: Highly parallel dbscan," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, New York, NY, USA, 2015, MLHPC '15, pp. 2:1–2:10, ACM.

[39] George Kollios, Dimitrios Gunopulos, Nick Koudas, and Stefan Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE Trans. on Knowl. and Data Eng.*, vol. 15, no. 5, pp. 1170–1187, Sept. 2003.

Part II

PAPERS